MicroBlaze Development Kit Tutorial

Conventions

This manual uses the following conventions. An example illustrates most conventions.

Typographical

The following conventions are used for all documents.

- `Courier font` indicates messages, prompts, and program files that the system displays.

  ```
  speed grade: - 100
  ```

- **`Courier bold`** indicates literal commands that you enter in a syntactical statement. However, braces “{ }” in Courier bold are not literal and square brackets “[ ]” in Courier bold are literal only in the case of bus specifications, such as bus [7:0].

  ```
      rpt_del_net=
  ```

  `Courier bold` also indicates commands that you select from a menu.

  `File` → **`Open`**

- *Italic font* denotes the following items.

  - ♦ Variables in a syntax statement for which you must supply values

    ```
    edif2ngd design_name
    ```

  - ♦ References to other manuals

    See the *Development System Reference Guide* for more information.

- ♦ Emphasis in text

  If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.

- Square brackets “[ ]” indicate an optional entry or parameter. However, in bus specifications, such as bus [7:0], they are required.

  `edif2ngd` `[`*`option_name`*`]` *`design_name`*

- Braces “{ }” enclose a list of items from which you must choose one or more.

  `lowpwr ={on|off}`

- A vertical bar “ | ” separates items in a list of choices.

  `lowpwr ={on|off}`

- A vertical ellipsis indicates repetitive material that has been omitted.

```
IOB #1: Name = QOUT’
IOB #2: Name = CLKIN’
.
.
.
```

- A horizontal ellipsis “…” indicates that an item can be repeated one or more times.

  `allow block` *block_name loc1 loc2 … locn;*

Online Document

The following conventions are used for online documents.

- Blue text indicates cross-references within a book. Red text indicates cross-references to other books. Click the colored text to jump to the specified cross-reference.
- Blue, underlined text indicates a Web site. Click the link to open the specified Web site. You must have a Web browser and internet connection to use this feature.

MDK Tutorial

This tutorial introduces you to designing with the MicroBlaze™ soft processor. It walks you through the steps required to build a MicroBlaze system using the MicroBlaze Development Kit (MDK) 2.2 flow and the Hello World tutorial design.

The Hello World design is provided to help you understand the basic steps in creating your own MicroBlaze system. The Hello World design is a simple design that only requires a JTAG_UART hardware peripheral. In this tutorial, this design is compiled with a debugging stub, the executable is loaded into memory, and the design is run through the hardware debugger to verify its functionality. To help you understand the basic steps in creating a MicroBlaze system, conceptual information is provided before each step is performed.

In this tutorial, you will learn how to do the following :

- Build both the hardware and software for the Hello World design
- Run the hardware and simulator debuggers on the Hello World design
- Simulate the Hello World design
- Use the ISE tools to implement and download the Hello World design

This tutorial includes the following sections:

System Requirements

Note MDK 2.2 does not support WebPACK.

The Hello World design runs on nearly all hardware systems.

You must be running iSE 4.2i and MDK 2.2.

This tutorial assumes that the target hardware board contains at least a Spartan-II, Spartan-IIE, Virtex, Virtex-E or Virtex-II device along with a JTAG interface.

MicroBlaze Development Tool Flow Overview

The MicroBlaze Development Tool (MDT) flow automates the MicroBlaze system building process. The following figure illustrates the steps in the MDT flow:

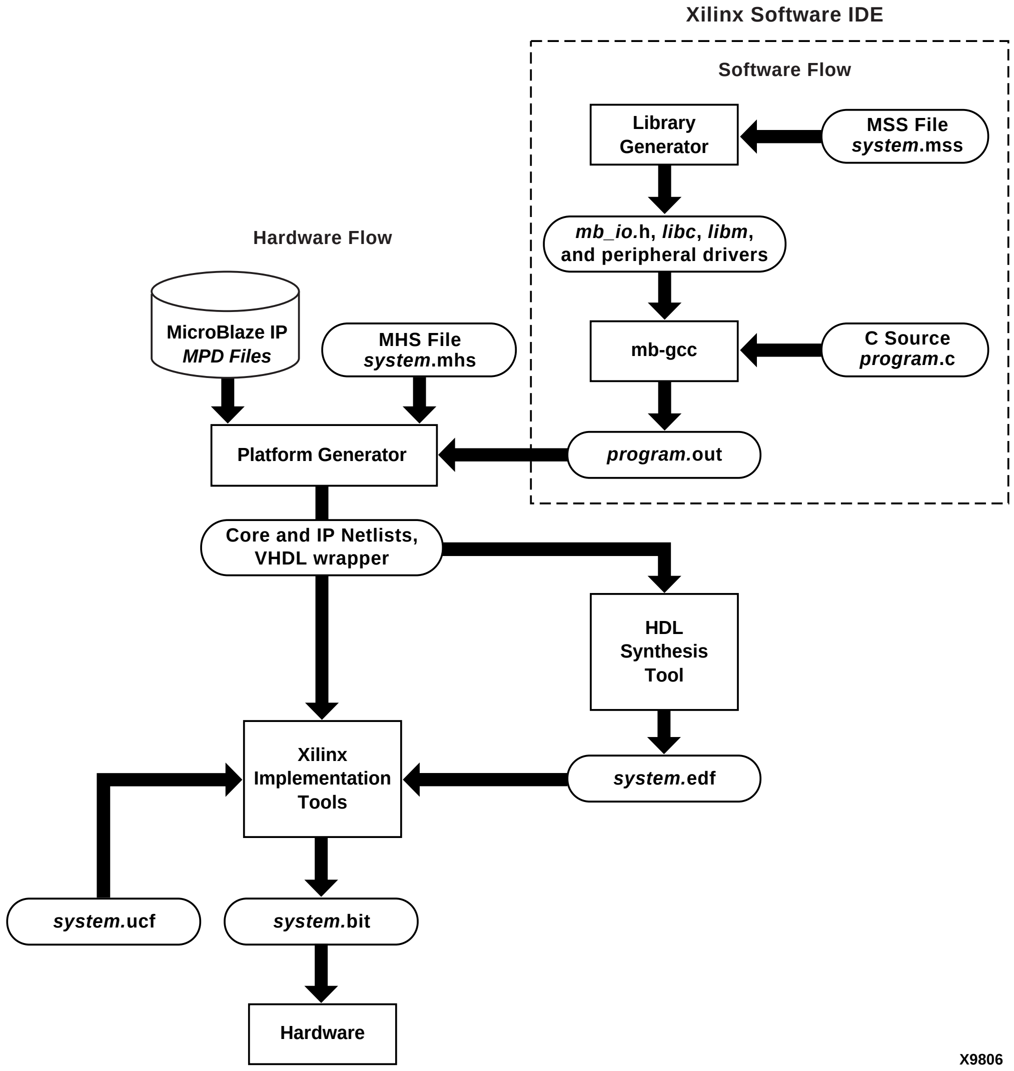


Figure 1-1 MDT Flow Diagram

The following steps provide a general description of the MDT flow:

1. Define your hardware and software systems using the Microprocessor Hardware Specification (MHS) and Microprocessor Software Specification (MSS) readable text files.
2. Use the hardware and software system definition files to build the MicroBlaze system automatically. This step includes integrating the MicroBlaze core and appropriate peripherals, and creating custom-built C libraries and drivers.

   The Platform Generator and Library Generator tools automatically set up the respective hardware and software for your system. The hardware and software flows are separate to allow for the hardware flow to take place as the software is being developed.
3. Once your system software is defined, use the Library Generator to build system-specific library C functions that map basic C functions (print, putnum, and so on) to peripherals and to configure the C libraries. Then use the mb-gcc compiler to compile your source code.
4. Use the Platform Generator to build the hardware files, which include the system netlists and HDL code, BlockRAM netlists initialized with the program code, and synthesis project files and simulation files.

Defining the Hello World Hardware

A MicroBlaze system is comprised of the following:

- MicroBlaze soft processor core
- On-chip block RAM
- Standard bus interconnects
- On-chip Peripheral Bus (OPB) peripherals

A MicroBlaze system can range from a processor core with a minimum of local memory to a large system with many MicroBlaze processors, sizable external memory, and numerous OPB peripherals. This section of the tutorial describes how to set up the MicroBlaze system hardware for the Hello World design.

Using the MHS File

The MicroBlaze hardware system is defined in a text-based Microprocessor Hardware Specification (MHS) file that is created by you. This file includes the following:

- Peripherals
- One of six MicroBlaze bus configurations
- System connectivity
- Address spacing

The MicroBlaze hardware is built around the On-chip Peripheral Bus (OPB), which is a CoreConnect™ peripheral bus standard developed by IBM. Each peripheral connected to the OPB is declared in the MHS file, including the MicroBlaze processor. Essentially the MicroBlaze processor is a master peripheral connected to the OPB.

Note For more information on the OPB standard, refer to the OPB Usage section of the *MicroBlaze Hardware Reference Guide*. For more information on the CoreConnect standard, see:

http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?iLanguageID=1&iCountry-ID=1&title=coreconnect

You can set all of your system hardware parameters in this file, including the following:

- Memory space for each peripheral (including MicroBlaze)
- IP version
- Interrupt connectivity and priority
- Peripheral-specific settings
- Signal connectivity.

Hello World MHS File

The MHS file in Figure 1-2 describes the hardware used in the Hello World design example. The Hello World system hardware contains a JTAG_UART peripheral and uses the MicroBlaze bus configuration 3.

To create the Hello World MHS file, perform the following steps:

1. Create a design directory on your system with the following name:

   Hello_World

2. Copy the contents of the MHS file in Figure 1-2 into a file with the following name, and save it to the Hello_World directory:

   system.mhs

```
SELECT BUS opb_v20
CSET attribute HW_VER = 1.00.b
CSET attribute INSTANCE = opb_bus
CSET signal SYS_Rst = sys_reset
END

SELECT SLAVE opb_jtag_uart
CSET attribute INSTANCE = myjtag
cset attribute HW_VER = 1.00.b
CSET attribute C_BASEADDR   = 0xFFFF8000
CSET attribute C_HIGHADDR   = 0xFFFF80ff
END

SELECT MASTER microblaze
CSET attribute INSTANCE = microblaze
cset attribute HW_VER = 1.00.b
CSET attribute CONFIGURATION = 3
CSET attribute C_LM_BASEADDR = 0x00000000
CSET attribute C_LM_HIGHADDR = 0x00000fff
END

SET signal SD = net_vcc
SET signal EN = net_gnd
```

Figure 1-2 Sample MHS file

For a complete listing of attributes and signals for each peripheral, review the MPD file associated with that peripheral. The MPD files are located here:

```
$MICROBLAZE/hw/coregen/ip/xilinx/microblaze_pcores/com/xilinx/ip2
processor/
```

Note For more information on the MPD and MHS files, refer to the MPD and MHS sections of the *MicroBlaze Hardware Reference Guide*.

Defining the Hello World Software

This section describes how to set up the MicroBlaze system software for the Hello World example using an MSS file.

Using Xilinx Microprocessor Software IDE (XSI) Utility

MDK 2.2 includes the Xilinx Microprocessor Software IDE (XSI) utility. This new utility provides an integrated graphical interface for creating the software system. From within XSI, you can run the software tools (such as the Library Generator and the GNU compiler tools) while specifying the software system (this is equivalent to building the MSS file). XSI requires a pre-designed MHS file as input. XSI provides the following:

- Editor and a project management interface for creating and editing source code
- Access to run Platform Generator tool, although it does not currently offer a complete hardware system design tool

To create a new XSI project, use the following steps:

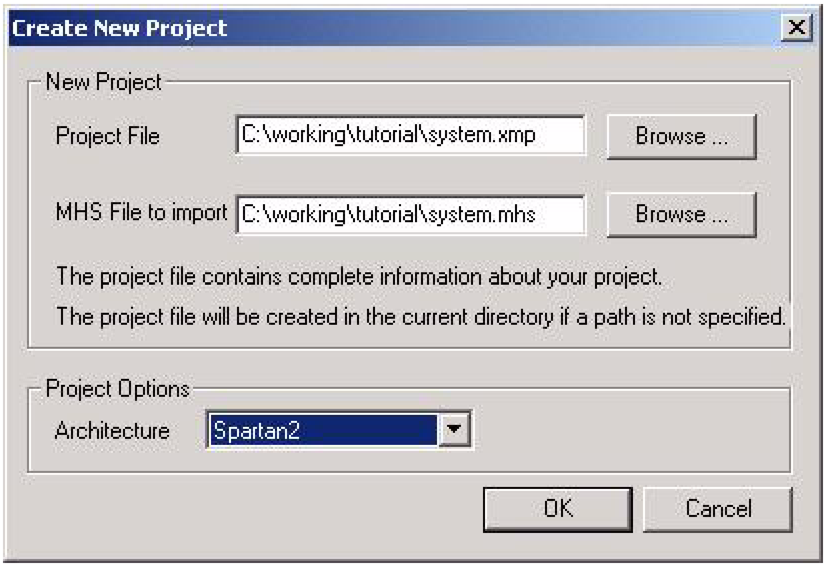
1. From the Windows Start Menu, open the XSI utility by selecting:

   `Programs → Xilinx MicroBlaze 2.2 → MicroBlaze IDE`

2. Using the MHS file created earlier, create a new project by selecting:

   `File → New Project`

   The Create New Project dialog box is displayed as shown in the following figure.



Project File: C:\working\tutorial\system.xmp

MHS File to import: C:\working\tutorial\system.mhs

The project file contains complete information about your project.

The project file will be created in the current directory if a path is not specified.

Architecture: Spartan2

Figure 1-3 Creating a New Project in XSI

3. In the MHS File to import field, use the Browse button to find the MHS file created earlier.
4. Name the project in the Project File field
5. Select the target architecture from the pull-down menu in the Architecture field.

Note If an MSS file already exists in the project directory, you are prompted to overwrite the MSS file. To save this file, rename it or move it to another directory because XSI will delete it. If you chose to use the existing MSS file, XSI assumes all of the MSS options set in that file.

6. Once the project is created, you can modify or create the source code. To add source code to the project from within XSI, select the following:

   `Project →Add Files`

   All headers and source files are listed in the project console on the left. Also, the hardware system is listed with the corresponding address space, as well as program options.

7. For the Hello World example, a simple C design is used that prints to STDOUT, as shown in the following figure:

```
#include <stdio.h>
main() {
  int i ;
  print("Hello World\n");
  putnum(i);
}
```

Figure 1-4 Hello World Example C Design

8. Create a new C source file by selecting:

   `File → New`

9. Copy the example Hello World C design or a similar one of your own. (It should output a character string using the “print” command or an integer using the “putnum” command.)

10. Create a /code directory in your Hello World design directory. XSI expects the file to reside in the /code directory so save the file in that directory.

11. Add the C source you just created to the project by selecting:

    `Project → Add Files`

    See the following figure.

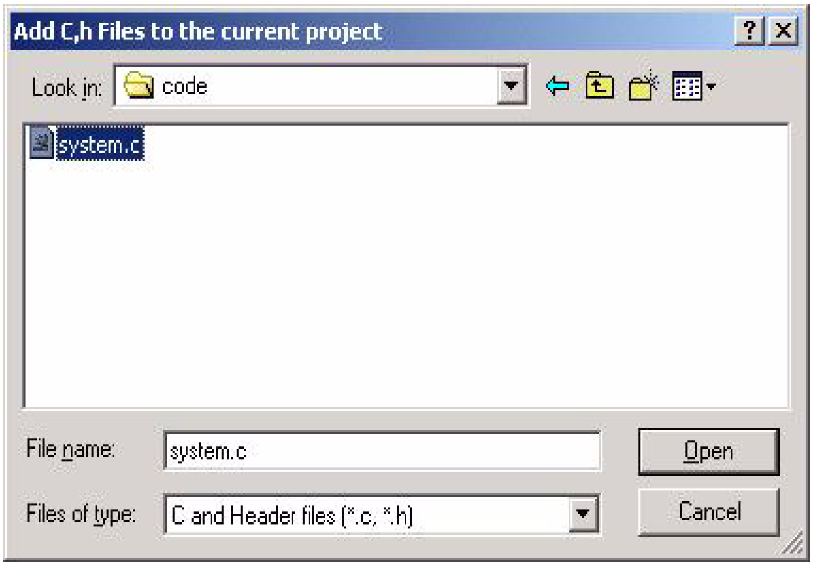

Add C,h Files to the current project
Look in:
code
system.c
File name:
system.c
Files of type:
C and Header files (*.c, *.h)
Open
Cancel

Figure 1-5 Adding C Source File in XSI

12. After the file is added, the source file appears in the Project View Window on the right under Sources. Also, the MHS settings appear under each peripheral (such as the BASE and HIGH address settings).

Note In the MDK 2.2 version of XSI, all of the hardware settings in the MHS file (peripheral base address, and so on) are not modifiable in XSI. To change these values, you can modify the MHS file outside of XSI and then reload it into the XSI project.

Defining the Hello World Design Software

Once the source has been created and added to the project, you can define the software system. From within XSI, you can set the Library Generator options as follows:

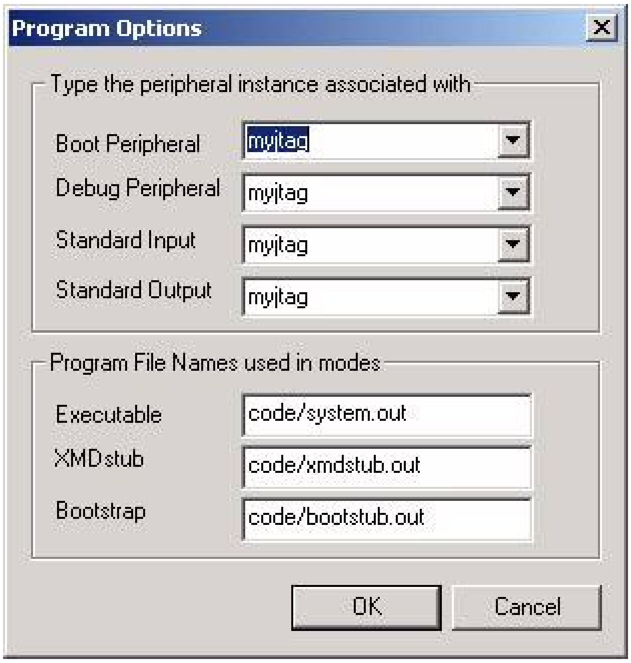
- Select STDIN / STDOUT peripherals
- Map peripherals to the appropriate drivers
- Set software attributes
- Set boot and debug options

- For peripherals that generate an interrupt, map the interrupt handler function to the peripheral

Note For more information on the Library Generator, refer to the Library Generator section of the *MicroBlaze Software Reference Guide.*

To set the Library Generator options in XSI, follow these steps:

1. To select the STDIN / STDOUT peripheral, double-click Program Options in the Project View Window. The Program Options dialog box is displayed as shown in the following figure.


Program Options
Type the peripheral instance associated with
Program File Names used in modes
OK
Cancel

Boot Peripheral: myjtag
Debug Peripheral: myjtag
Standard Input: myjtag
Standard Output: myjtag
Executable: code/system.out
XMDstub: code/xmdstub.out
Bootstrap: code/bootstub.out

Figure 1-6 Setting Program Options

You can use this dialog box to select peripherals for the following:

- Boot Peripheral
- Debug Peripheral
- Standard Input
- Standard Output

You can also select the Executable, XMDstub and Bootstrap files.

2. In the Program Options dialog box, select myjtag for the Boot and Debug Peripheral and for Standard Input and Standard Output.

Peripheral Options

In the Peripheral Options window, the driver and driver version are selectable. To access the peripheral options, double-click the peripheral name under System BSP in the Project View window.

If the peripheral is capable of generating an interrupt, you can enter the name of the Interrupt Handler. The peripheral must have the interrupt signal set in the MHS file for an Interrupt Handler to be selectable in XSI. Otherwise, the Peripheral Configuration section of the Peripheral Options window is greyed out. Interrupts are not used in the Hello World example. No changes from the default settings need to be made for the Hello World design.

Setting the Hello World XSI Project Mode

You can run a MicroBlaze project in one of the following modes:

- Executable

  This is the default mode. Use this mode to generate a stand-alone executable program.

- Bootstrap

  Use this mode when to use a bootstrap program to load your program.

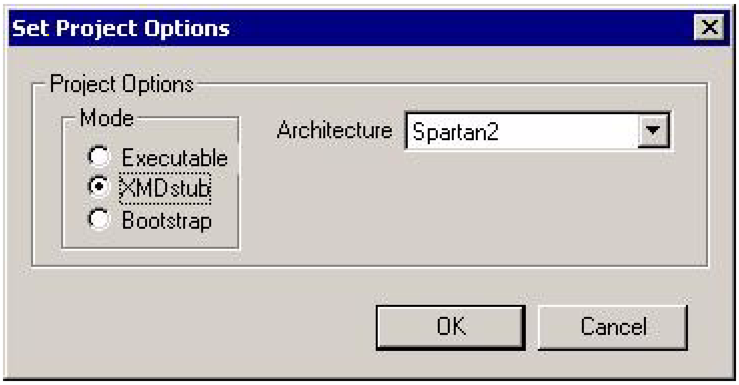
- XMDstub

  Use this mode to run debugging on your hardware target.

1. Use the following command to select your project mode in XSI:

   `Project → Set Project Options`

   The Set Project Options dialog box is displayed as shown in the following figure.



Mode: Executable [✓ XMDstub] Bootstrap

Architecture: Spartan2

Figure 1-7 Setting Project Options

2. Set the Mode to **`XMDstub`**, and click **`OK`**.

Building the Hello World Design Drivers and C Libraries

The next step in building your Hello World system is using the Library Generator to create the system drivers and C libraries.

The Library Generator is invoked through XSI and performs the following functions:

- Compiles Peripheral Drivers

  The Library Generator searches the $MICROBLAZE/driver directory for driver source files using the DRIVER attribute for the directory name. In the driver directory are the C driver source, header files, and a makefile for the driver. The Library Generator then copies this directory over to the $MICROBLAZE_PROJECT/libsrc directory and runs a makefile to compile the drivers.

  For the Hello World design, the JTAG_UART drivers are copied over into the project /lib directory.

- Creates Header File

  The Library Generator creates the mbio.h header file and places it in the $MICROBLAZE_PROJECT /include directory. This file contains the base address and interrupt masks for each peripheral.

For the Hello World design, the JTAG_UART base addresses are added to the mbio.h file.

- Sets up STDIN / STDOUT

  The Library Generator sets up the STDIN / STDOUT for your system using the STDIN and STDOUT attributes in the MSS file and the INBYTE and OUTBYTE attributes in the Microprocessor Peripheral Definition (MPD) file. A peripheral can only be used for system STDIN / STDOUT if the corresponding INBYTE / OUTBYTE attributes are set in the MPD file.

  If a peripheral is set with STDIN / STDOUT attributes, the Library Generator uses the peripheral's inbyte.c and outbyte.c functions in the C libraries. Consequently, any function that is dependent on inbyte() or outbyte() functions (such as print, get, printnum, and so on) maps to that peripheral. If the base address of any peripheral is changed, the Library Generator must be run again.

  For the Hello World design, the JTAG_UART peripheral is mapped to Standard Input and Standard Output.

- Maps Interrupt Routines

  The Library Generator maps interrupt routines to a particular interrupt signal. In the MSS file, the INT_HANDLER attribute allows an interrupt handler routine to be associated with an interrupt signal. The Library Generator uses this attribute to configure the interrupt controller (in the case of multiple interrupts) to call the appropriate interrupt routines on an interrupt. If INT_HANDLER attribute is not specified, the Library Generator uses a default dummy handler routine for that interrupt.

  Interrupt Routines are not used in the Hello World design.

Running Library Generator in XSI

Note For more information on running the Library Generator, refer to the Library Generator section of the *MicroBlaze Software Reference Guide*.

To run the Library Generator on the Hello World design in XSI, select the following:

`Run` → **`Library Generator`**

The results appear in the Console Window.

Note The current version of XSI does not check for dependencies. You are responsible for invoking programs in the proper order.

Compiling the Hello World Code

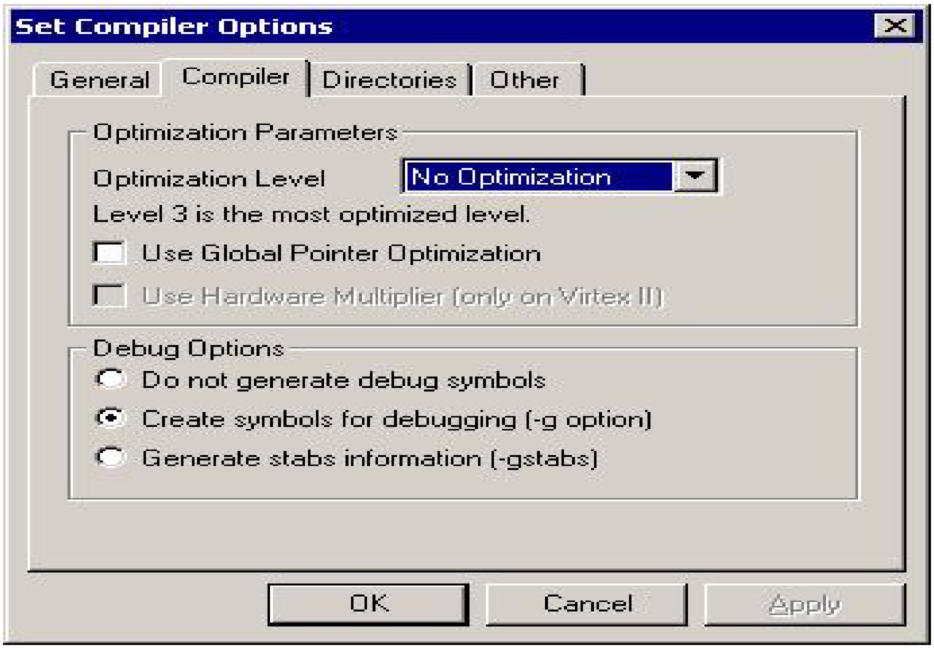
After the drivers and C libraries are generated, the next step is to compile the source code using the MicroBlaze GNU tools. These tools are very similar to the standard GNU toolset. The MicroBlaze GNU tools include the top-level GNU program mb-gcc, which calls out the compiler, the mb-as assembler, and the mb-ld linker/loader. You invoke the GNU tools through XSI.

Running GNU Tools in XSI

1. To select GNU compiler options, select the following:

   `Run` → **`Select Options`** → **`Compiler Options`**

The Set Compiler Options dialog box is displayed as shown here:



Optimization Parameters

Optimization Level: No Optimization

Level 3 is the most optimized level.

- [ ] Use Global Pointer Optimization
- [ ] Use Hardware Multiplier (only on Virtex II)

Debug Options

- ○ Do not generate debug symbols
- ● Create symbols for debugging (-g option)
- ○ Generate stabs information (-gstabs)

Figure 1-8 Setting Compiler Options

2. Browse through the compiler options.
3. Under the Compiler tab, set the Debug Options to Create symbols for debugging (-g option). This option inserts debug information into the executable file.
4. Compile the Hello World code by selecting:

   `Run` $\rightarrow$ **`Compiler`**

   The results appear in the Console Window.

Note For more information on these and other options for the GNU tools, see the Software Application Development Tools - GNU Compiler Tools section of the *MicroBlaze Software Reference Guide.*

Building the Hello World Hardware System using Platform Generator

After the MHS file is created and the software is compiled using the Library Generator and GNU tools, the next step is to combine the hardware and software flows. The Platform Generator uses the MHS system description to construct the following hardware files for the MicroBlaze system:

- IP and MicroBlaze netlists
- BlockRAM memories configured with the program information
- Synthesis project files
- Simulation HDL files

The Platform Generator then ties the system together by generating a flattened netlist, or by creating it hierarchically using VHDL wrapper files. You can invoke the Platform Generator in XSI.

Running Platform Generator in XSI

To run Platform Generator on the Hello World design in XSI, select the following:

Run → XM

The results appear in the Console Window in XSI.

The XSI call to Platform Generator assumes the “-flat” and “-s 2” options. The “-flat” option specifies that a flattened EDIF netlist of the Hello World design is produced. The “-s 2” option specifies that XST is used as the synthesis tool. In the MDK 2.2 version of XSI, Platform Generator is only partially supported. If you want to change these options, you can either run Platform Generator from the command line or modify the Platform Generator section of the *flow mode*.opt file in the XSI project directory.

The current version of XSI does not check for flow dependencies. Consequently, if a change is made in one part of the flow that affects later steps in the flow, you must run all steps again.

Note For more information on using the Platform Generator tool, see the Microprocessor Development Tools (MDT) Flow - Platform Generator section of the *MicroBlaze Software Reference Guide*.

Implementing the Hello World Design

Once a netlist for the Hello World design is created, it can be implemented in the ISE tools and downloaded to the device.

Note For more information on using ISE, refer to the ISE online help available from the Help menu within Project Navigator, and to the online ISE 4 User Guide at:

http://toolbox.xilinx.com/docsan/xilinx4/pdf/docs/xug/xug.pdf

1. Open the ISE Project Navigator and start a new Project by selecting:

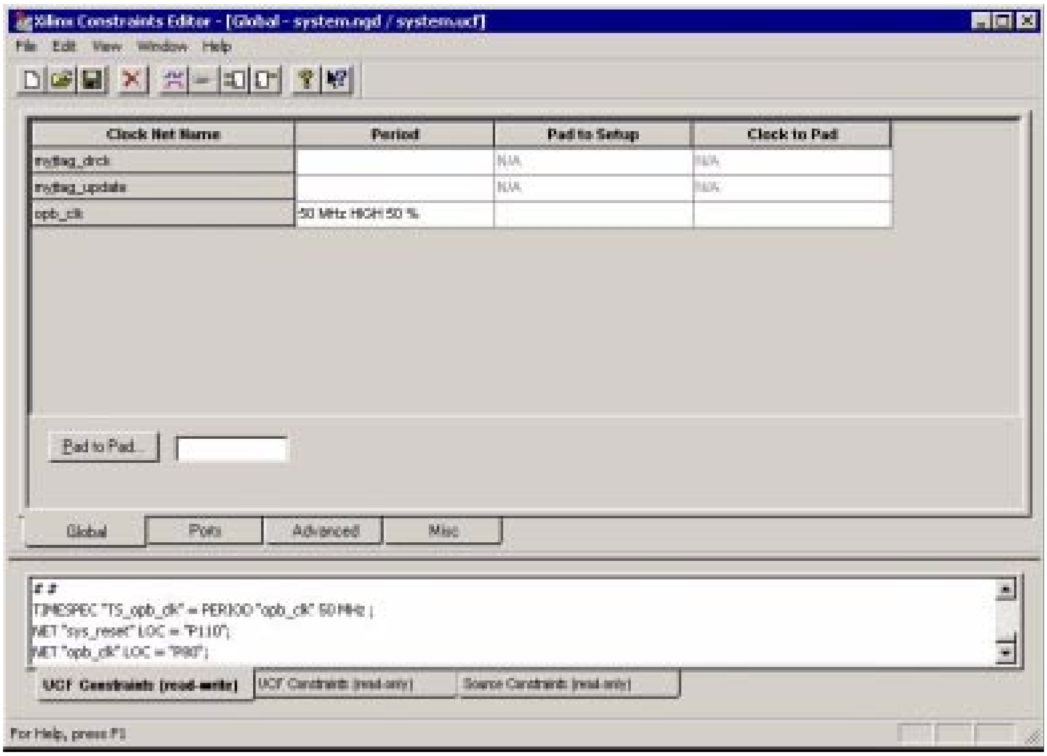
   `File →New Project`

2. Select the appropriate Device Family, Device, and Speed Grade settings.
3. Select the EDIF design flow and name the project.
4. Add the Hello World design netlist by selecting:

   `Project →Add Source`

5. Browse to the netlist generated from XSI (all netlists generated by Platform Generator reside in the $MICROBLAZE_PROJECT/ implementation directory.) Add the netlist to your project. Note that it is now listed under the Sources in Project window on the upper right-hand side.
6. Before you can implement the design, you must enter a few constraints as follows:

   a) Within the Project Navigator, you can launch the Xilinx Constraints Editor from the Processes window. You must have a design file selected in the Sources window. Then double-click Constraints Editor in the Processes window, which is located within User Constraints underneath Design Utilities. The Constraints Editor is shown in the following figure.


Xilinx Constraints Editor - [Global - system.ngd / system.ucf]
File Edit View Window Help
Pad to Pad...
Global
Ports
Advanced
Misc
UCF Constraints (read-write)
UCF Constraints (read-only)
Source Constraints (read-only)
For Help, press F1

Clock Net Name	Period	Pad to Setup	Clock to Pad
mytag_drck		N/A	N/A
mytag_update		N/A	N/A
opb_clk	50 MHz HIGH 50 %		

```
# #
TIMESPEC "TS_opb_clk" = PERIOD "opb_clk" 50 MHz ;
NET "sys_reset" LOC = "P110";
NET "opb_clk" LOC = "P80";
```

Figure 1-9 Entering Clock Speed Constraints

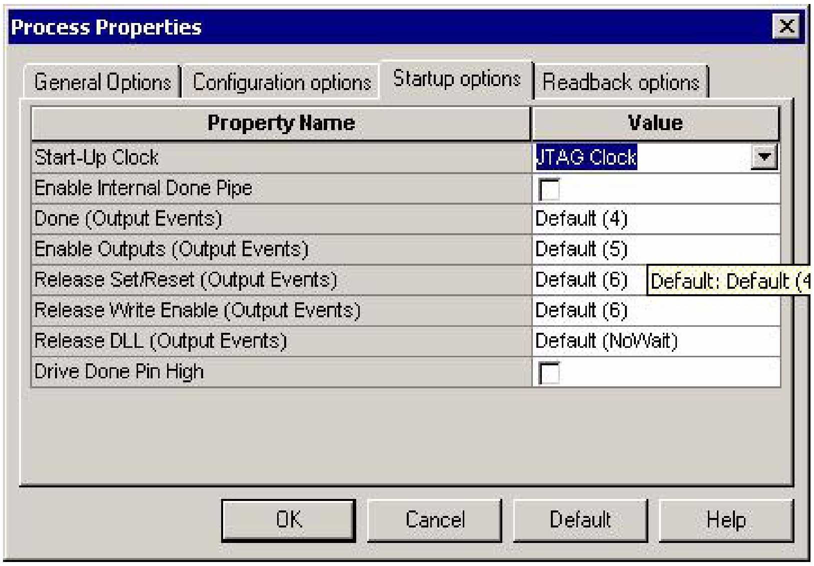
b) Specify a timing constraint for the system clock that matches the speed of the board clock. Check the documentation for your board for the board clock speed.

c) Specify the OPB_CLK and SYS_RESET pins in the design. The SYS_RESET on MicroBlaze is active high and *must* be connected to an appropriate driver. Check the documentation for your board for the clock and reset pins.

d) Save the constraints and return to the Project Navigator.

7. To change StartUp Clock to JTAG CLK, perform these steps:

   a) Right click on Generate Programming File in the Processes Window, and select Properties to display the Process Properties dialog box.



Property Name	Value
Start-Up Clock	JTAG Clock
Enable Internal Done Pipe	☐
Done (Output Events)	Default (4)
Enable Outputs (Output Events)	Default (5)
Release Set/Reset (Output Events)	Default (6)
Release Write Enable (Output Events)	Default (6)
Release DLL (Output Events)	Default (NoWait)
Drive Done Pin High	☐

Default: Default (4

Figure 1-10 Setting Process Properties

   b) Select the Startup Options tab. Change the Startup Clock to JTAG Clock.

8. Implement the design by double-clicking Implement Design in the Processes Window.

Configuring the Hello World Design

Once the Hello World design is implemented, you can download it to the device.

1. Open the Xilinx iMPACT tool by double-clicking Configure Device (iMPACT) under Generate Programming File in the Project Navigator.

2. If the Boundary Scan chain is detected, proceed to the next step. If not, add devices to the chain from within iMPACT by selecting:

   Edit → Add Device

3. Browse to the device’s BSDL file. Repeat for as many devices as there are in the chain.
4. Right-click on the target device and select Program to program the device.

Debugging the Hello World Design

Now that the target device is programmed with the Hello World design, you can interface to the board through the JTAG_UART peripheral.

The MicroBlaze Development Tools include a GNU software debugger. This debugger (mb-gdb) supports two debug targets: a cycle-accurate Instruction Set Simulator (ISS) or a hardware board.

The Xilinx Microprocessor Debug (XMD) engine provides a unified interface between the debugger and the debug targets. XMD must be run in conjunction with the debugger and provides communication with the target hardware in hardware debugging (it appears as a shell window running in the background.) If the hardware board is your debug target, a debug stub (XMDstub) must be running on the target board. This stub should be already running on the target board because the XMDstub program option was set in XSI.

Software Debugging Using the Cycle Accurate ISS

Assuming that the previous steps were run successfully and the Hello World code was compiled using the –g option, hardware debugging is ready to be set up.

1. Verify that the Hello World code was compiled with the –g option by checking the xflow.log file generated by XSI earlier. Examine the line that contains the mb-gcc command. It should indicate that the -g option was specified when compiling the source. If not, go back to the Project Mode section of this tutorial and change the Project Mode to XMDstub, and then rerun the entire software flow.
2. Open a Xygwin shell by selecting:

   `Programs → Xilinx MicroBlaze 2.2 → Xygwin Shell`

3. Run XMD using a simulator debug target as shown in the following example:

   ```
   xmd -t sim -u gdb
   MicroBlaze XMD Engine
   Using Simulator
   Use the following command in GDB to connect:
   target remote host_name:1234
   ```

4. From the Windows Start menu, open the debugger by selecting:

   Programs → Xilinx MicroBlaze 2.2 → MicroBlaze GNU Debugger

5. In mb-gdb, load the compiled source code (system.out.executable) for debugging by selecting:

   File → Open

   The compiled source is located in the code directory within the Hello World directory.

6. From the Run menu, select Connect to Target in the mb-gdb window.

7. In the Target Selection dialog box, select the following:

   Target:Remote/TCP

   Hostname:localhost

   Port:1234

   You can now use the mb-gdb interface to debug your code running on a remote hardware target. The XMD console reports the status on the connection of the debugger and XMDstub. Verify that communication is established.

8. You can execute the program by single-stepping (Control → Step) or executing the entire program (Control → Continue). The STDOUT from your program appears in the XMD console.

Note See the Debug Tool Chain section in the *MicroBlaze Software Reference Guide* for more information on using the debugger.

Hardware Target Debugging

Assuming that the previous steps in the Hello World tutorial were run successfully and the XMDstub program is running on your target hardware, hardware debugging is ready to be set up.

1. Verify that the Hello World design was created in XMDstub mode by checking the xflow.log file generated by XSI earlier. Examine the mode settings for Library Generator and Platform Generator. It should indicate that the XMDstub mode was specified. If not, go back to the Project Mode section of this tutorial and change the Project Mode to XMDstub, and then rerun the entire flow.

2. Open a Xygwin shell by selecting:

   Programs → Xilinx MicroBlaze 2.2 → Xygwin Shell

3. Run XMD using a hardware debug target as shown in the following example:

   ```
   xmd -t hw -j 2 -u gdb

   MicroBlaze XMD Engine

   Using Hardware board debugging through XMD stub

   Connecting to XMD stub at baud rate: 19200 bps

   XMD stub initialized. Version No: 1

   Use the following command in GDB to connect:

   target remote host_name:1234
   ```

4. From the Windows Start menu, open the debugger by selecting:

   Programs → Xilinx MicroBlaze 2.2 → MicroBlaze GNU Debugger

5. In mb-gdb, load the compiled source code (system.out.xmdstub) for debugging by selecting:

   File → Open

   The compiled source is located in the code directory within the Hello World directory.

6. From the Run menu, select Connect to Target in the mb-gdb window.
7. In the Target Selection dialog box, select the following:

   `Target:Remote/TCP`

   `Hostname:localhost`

   `Port:1234`

   You can now use the mb-gdb interface to debug your code running on a remote hardware target. The XMD console reports the status on the connection of the debugger and XMDstub. Verify that communication is established.
8. Execute the program either by single stepping (Control > Step) or execute the entire program (Control > continue). You should see the STDOUT from your program appear in the XMD console.

Note See the Debug Tool Chain section in the *MicroBlaze Software Reference Guide* for more information on using the debugger.

Simulating the Hello World MicroBlaze System

This section describes the steps necessary to complete both functional and timing simulation using the ModelTech ModelSim simulator. Other simulators are supported, however, the .do script created by the Platform Generator is only supported by ModelSim.

Because the Hello World example communicates via JTAG ports that cannot be simulated, the ASCII Hello World characters are not visible in simulation. However, the instruction executable is visible.

Note For more information on using ModelSim, see the ModelSim section in the ISE 4.1 tutorial available online at:

ftp://ftp.xilinx.com/pub/documentation/ise4_tutorials/ise4tut.pdf

Functional Simulation of the Hello World Design

The following steps describe how to perform a functional simulation of the Hello World design:

1. Before performing a simulation, run both the Library Generator and the mb-gcc compiler in executable mode. Go back to the "Setting the Hello World XSI Project Mode" section and select the

Executable mode. Run both Library Generator and the mb-gcc compiler from within XSI again.

2. Run Platform Generator command line from within the Hello World project directory as follows:

```
platgen -mode executable -a spartan2 –flat –sim 1 -mss system.mss
system.mhs
```

Platform Generator creates the following files when run with the -sim 1 option and places them in the $MICROBLAZE_PROJECT/simulation directory.

- platgen -mode executable -sim 1 system.mhs
- ModelSim .do file (system.do)
- Top-level (system.vhd) HDL file
- Memory (local_memory.vhd) HDL file (if not run in –flat mode)
- Simulation models for any peripherals used (if not run in –flat mode)

Compiling the Hello World Design in ModelSim

To compile your design in ModelSim, perform the following steps:

1. Open ModelSim.
2. At the ModelSim command prompt, go to the Hello World simulation directory.
3. At the ModelSim command prompt, enter:

   do *system*.do

   The script runs and the necessary libraries are compiled. HDL files from the $MICROBLAZE/hw directory are compiled into the simulation directory.

The following table provides a list of the libraries typically compiled.

Table 1-1 Simulation Libraries

Library	Description
common	Components used in peripherals and in the MicroBlaze core (for example, pselect, or_gate, and so on)
microblaze_x	Simulation model for the MicroBlaze core (x is a number from 1 to 6 that corresponds to the version of the MicroBlaze core that is used)
opb_v20	On-chip Peripheral Bus component
work	Contains the compiled top-level and memory files

Note Additionally, a library for any custom peripherals is created.

Running the Simulation

Follow these steps to simulate your design:

1. At the command prompt, enter:

   ```
   vsim system
   ```

   In this case, design_name is system.

   Note If a test bench has been created, you must load it at this point.

2. At the very minimum, you can exercise the Hello World system using clock and reset stimuli. At the ModelSim command prompt, enter:

   ```
   force opb_clk 0 0ns, 1 50ns -r 100ns
   force sys_reset 0 0ns, 1 300ns, 0 1us
   ```

   Note 50 ns represents the active high portion of the clock waveform, and 100 ns is the period of the clock.

3. Display the windows, add signals to the wave, and start the simulation. At the ModelSim command prompt, enter:

   ```
   view wave structure signals
   add wave *
   run 5 us
   ```

To speed up simulation iterations, you do not need to run the .do file repeatedly. If only the C program has changed, then you can simply recompile the top-level and memory files (peripherals and processor core are unchanged). You can extract the necessary commands from the .do file and run the simulation from the command line.

Timing Simulation

To perform a timing simulation, your design must first be implemented using the Xilinx ISE tools. The flow is the same as for any Xilinx design and is described in the online software manuals accessible at:

http://support.xilinx.com

Potential Pitfalls

Due to system differences, you may encounter errors may occur while performing this tutorial. Refer to the following list of problems and solutions to help you resolve any errors:

Problem

When running XM from within XSI, the following error occurs:

```
"the system.c file is not found in the code directory"
```

Solution

Make sure source code is the /code directory.

Problem

When running XMD, the following error occurs:

```
C:\>xmd -t hw -j 2 -u gdb -V
MicroBlaze XMD Engine
Using Hardware board debugging through XMD stub
AutoDetecting cable. Please wait.
Connecting to cable (USB Port).
Cable connection failed.
Connecting to cable (Parallel Port - LPT1).
 Driver Version = 503.
Cable connection failed.
```

```
Connecting to cable (Parallel Port - LPT2).
 Driver Version = 503.
Cable connection failed.
Connecting to cable (Parallel Port - LPT3).
 Driver Version = 503.
Cable connection failed.
```

Solution

Make sure iMPACT is closed while running XMD.

Problem

While constructing platform, the following errors occur:

```
Checking address space for memory peripherals …
ERROR: address space of lmb_bram_0_i needs to be of
   a fixed size!
Platform build failed
ERROR:Xflow – Program platgen returned error code
   255. Aborting flow execution…
```

Solution

A larger memory space is needed for Virtex2 devices; small space is not supported in Platform Generator. You must increase the C_LM_HIGHADDR value in the MHS file to 0x00001fff.

Revision History

Table 1-2 Revision History

Date	Version	Revision
10/25/01	1.9	Initial MDK tutorial. Covered C2bits
1/21/02	2.1	Updated MDK tutorial with 2.1 information
2/15/02	2.1.1	Made changes to Make section, Platform Generator
4/02	2.2	Updated for MDK 2.2. Added information on the XSI utility.